BISS Biodiversity Information Science and Standards

Biodiversity Information Science and Standards 3: e39150
doi: 10.3897/biss.3.39150

OPEN ACCESS

Conference Abstract

Notebook: Customizable web forms for recording observations

Olli Raitio[‡], Aino Juslén[§], Päivi Sirkiä[§], Aleksi Lehikoinen[§], Marko Tähtinen[|], Esko Piirainen[‡], Ville-Matti Riihikoski[‡]

‡ Biodiversity Informatics Unit, Finnish Museum of Natural History, Helsinki, Finland
§ Zoology Unit, Finnish Museum of Natural History, Helsinki, Finland
| Former member of Biodiversity Informatics Unit, Finnish Museum of Natural History, Espoo, Finland

Corresponding author: Olli Raitio (olli.raitio@helsinki.fi)

Received: 16 Aug 2019 | Published: 20 Aug 2019

Citation: Raitio O, Juslén A, Sirkiä P, Lehikoinen A, Tähtinen M, Piirainen E, Riihikoski V-M (2019) Notebook: Customizable web forms for recording observations. Biodiversity Information Science and Standards 3: e39150. https://doi.org/10.3897/biss.3.39150

Abstract

'Notebook' is one of the primary data management systems of the Finnish Biodiversity Information Facility (FinBIF). It is a web solution for recording opportunistic as well as sampling-event-based species observations. It is being used for systematic monitoring schemes, various citizen science projects, and platforms for species enthusiasts. Notebook's main software component is LajiForm, which is the engine that renders a given JSON Schema into a web form. LajiForm is a separate, reusable module that is fully independent from other FinBIF systems. Notebook as a whole, includes other features embedded in FinBIF, such as linking users' geographical data to observation documents, spreadsheet document importing and form templates. We will demonstrate how the Notebook system works as a whole and also focus on LajiForm's technical aspects (Fig. 1).

All Notebook forms use FinBIF's ontological schema in JSON Schema format. Rendering user-friendly web forms based on a single schema is a difficult task, because the web form should be asking meaningful questions, instead of just rendering the schema fields according to the form description. We want to present questions in an interactive style. For instance, after drawing a geographical location on a map for a potential flying squirrel nesting tree, we would ask "did you see droppings at the nest?", and answering "yes"

© Raitio O et al. This is an open access article distributed under the terms of the Creative Commons Attribution License (CC BY 4.0), which permits unrestricted use, distribution, and reproduction in any medium, provided the original author and source are credited.

would update the document to include a flying squirrel taxon identification with fields "breeding" and "record basis" filled in but not rendered to the form. A simpler form engine without a user interface (UI) customization layer would just render the "taxon", "breeding" and "record basis" fields and the user would have no understanding why there are so many fields to answer and how they relate to their work or study. Some forms are complex, e.g., for experienced biology enthusiasts who need a form that is advanced, customizable, and compact. Some forms are simple, e.g., for elementary school children.

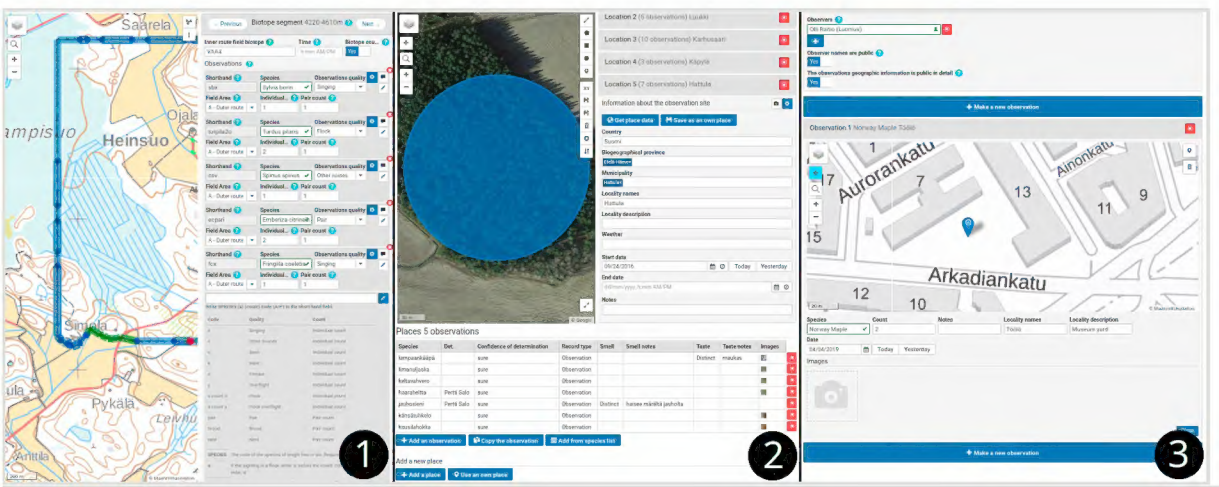

Figure 1.

Examples of forms rendered by LajiForm. 1: Bird line transect study form. 2: Mushroom atlas citizen science project form. 3: A simple form for a project for elementary school children.

To tackle these challenges, LajiForm uses a separate schema for UI that allows everything from simple customization like:

1. defining widgets for fields (e.g., date widgets, taxon autocomplete widget, map widget),
2. changing field order or
3. customizing field labels; to more complex customization like
4. transforming the schema object structure,
5. defining conditions when certain fields are shown or
6. if updating a field should have an effect on other fields.

All the functionality is split into a loosely coupled collection of components, which can be either used as standalone components or composed together in order to achieve more advanced customization. The programming philosophy has drawn inspiration from functional programming, which has been helpful in writing isolated, composable functionality.

LajiForm is written with the JavaScript framework React. LajiForm is built on top of react-jsonschema-form (RJSF), which is an open source JSON schema web form library provided by Mozilla. RJSF handles only simple customization, but it is very flexible in design and allows us to build extensions with features that are more powerful. Some features and design proposals were submitted to Mozilla – FinBIF is the largest code contributor to RJSF outside of Mozilla, with a dozen pull requests merged.

Keywords

observation recording, web forms, JSON Schema, web development, JavaScript, React

Presenting author

Olli Raitio

Presented at

Biodiversity_Next 2019